Agile Decision Agent for Service-Oriented E-Commerce Systems

Jiyong Zhang
HCI Group,
Ecole Polytechnique Fédérale
de Lausanne (EPFL),
CH-1015, Switzerland
Email: jiyong.zhang@epfl.ch

Pearl Pu
HCI Group,
Ecole Polytechnique Fédérale
de Lausanne (EPFL),
CH-1015, Switzerland
Email: pearl.pu@epfl.ch

Boi Faltings
AI Laboratory,
Ecole Polytechnique Fédérale
de Lausanne (EPFL),
CH-1015, Switzerland
Email: boi.faltings@epfl.ch

Abstract

In a service-oriented e-commerce environment, it is a crucial task to help consumers choose desired products efficiently from a huge amount of dynamically configured product candidates. Decision agents can be designed to provide interactive decision aids for end-users by eliciting their preferences and then recommending matching products. In reality the users' preferences may keep changing along with the dynamic decision environment and may not be fully satisfied. As a result, the decision agent is required to be agile*: it should allow decision making with an incomplete user's preference model and should afford users to add, retract or revise their various preferences with little effort. In this paper we propose the general design of an agile decision agent to meet this need. We model users' preferences with the soft constraint technique and elicit them by the example critiquing interaction paradigm.*

1 Introduction

A service-oriented e-commerce system could provide a huge amount of product information by dynamically integrating various external web services together to meet the consumers' diverse requirements. For instance, by utilizing Amazon e-commerce web services[1], an online website can be implemented to allow consumers to access information on millions of products provided by Amazon Company. Thanks to the new merging technology of XML and web services, a user is now able to access the structured product data such as travel plan offers from multiple sources of web services or their combinations. The product information might also be changed dynamically while the user is making choices. As a result, the user is facing a much more complicated decision problem than in the static situation.

Studies from economics and psychology have shown that the individuals only have *bounded rationality* when making decisions due to their limited knowledge and computational capacities[20], it is quite unpractical for buyers to evaluate all available products one by one in depth while making purchase decisions if no decision aid tool is provided. Therefore, a crucial task for service-oriented e-commerce systems, where the products are potentially changing within the decision processes, is to provide effective product search and decision aid tools to help buyers choose the preferred products with a reasonable amount of effort and time. Fortunately, web services make product data available in structured form that can be searched by machine.

It has been found that decision agents can help end-users effectively make shopping decisions in online shopping environments [7][19]. Currently many existing e-commerce applications *elicit* users' preferences by asking a series of fixed questions and then return a set of possible choices based on the answers. For example, a popular travel planning website, Travelocity[2], asks each user several questions about the itinerary and airline preferences, and then shows a list of possible solutions to the user. Such elicitation processes implicitly assume that users have a pre-existing and stable set of preferences.

However, this assumption has been challenged by the studies of actual decision makers' behavior from behavioral decision theory[13][4] and these studies have pointed out the adaptive and constructive nature of human decision making. Specifically, a rigid preference elicitation procedure can cause severe problems in the following situations:

- Users are not aware of all preferences until they see them violated. For example, a user does not think of stating a preference for intermediate airports until a solution includes a change of flight in a place that he or she dislikes.

[1] see http://www.amazon.com/webservices.

[2] see http://www.travelocity.com/.

- Elicitation questions that do not concern the user's true objective can force him or her to formulate *means objectives* corresponding to the question. This phenomenon has been studied by Keeney[9] in his work on value-focused thinking.

- Preferences are often in contradiction and require the user to make tradeoffs. Users may add, remove or change their preferences in an arbitrary order. Again, this is not supported by a tool with a rigid preference elicitation procedure.

- In the service-oriented environment, where the product data are provided from external web services and may be changed unexpectedly during the decision process, a strictly system-driven elicitation procedure may lead a user to express some useless preferences while neglecting some truly useful preferences. For example, a travel planning system with such rigid manner may ask a user to select an intermediate airport due to the fact that no direct flight is available. Later on if the system has a new non-stop flight which satisfies the user's true preferences better, this new choice may never be found.

To support these properties of human decision making and the dynamic nature of service-oriented e-commerce systems, the decision agent is required to be *agile*. It should support incremental construction, revision and tradeoffs of the user's preferences. More precisely, we define agility as the ability to satisfy the following two requirements: 1)allow decision making with an incomplete preference model that only captures those preferences that have actually been expressed by the user; 2)allow users to add, retract and revise preferences in any order during the decision process.

In this paper, we propose the general approach of designing an agile decision agent to meet the requirements of service-oriented e-commerce systems. The agility of the proposed decision agent lies in the following two aspects. First, the interaction manner between the agent and the user is based on the example-critiquing interaction paradigm which supports incremental preference construction and tradeoff. Second, we model user's preferences with the soft constraint technique which allows users to add/modify preferences in any order they desire. To the best of our knowledge, our work is the first one to address the agility issue of e-commerce systems.

This paper is organized as follows: in Section 2 we introduce the related work. Section 3 describes the structure of the service-oriented e-commerce system with the agile decision agent. The method of constructing an agile preference model with the soft constraint technique is studied in the following section. Discussions and conclusions are given in Section 5 and 6, respectively.

2 Related Work

2.1 Service-Oriented E-Commerce Systems

Web Services (WS) are encapsulated, platform- and network-independent operations that are accessible to other applications or end-users. Web services have an interface described in a machine-processable format by the web services description language (WSDL)[5]. This description allows an application to dynamically determine a web service's operations, parameters, and return values. More importantly, web services provide data in structural form such as XML and thus provide an ideal approach for developing e-commerce systems combining various data sources to satisfy various users' diverse and complex shopping needs. However, many researchers currently mainly focus on the development of the service-oriented infrastructure such as service discovery and composition while the issue of alleviating the users' burden of finding the most satisfying product from a large product depository is largely neglected. As we mentioned earlier, an agile decision agent is indispensable in such systems to help users find the desired products.

2.2 Methods of Preferences Elicitation and Representation

One potential method of modeling user's preferences is based on multi-attribute utility theory (MAUT)[10]. MAUT is a powerful and widely used approach for solving decision problems in many domains including e-commerce applications. For example, Stolze [21] proposed the scoring tree method for building interactive e-commerce system based on multi-attribute utility theory (MAUT). However, The MAUT approach generally requires a specific assumption to limit the complexity of the utility function and such assumption is often violated in many real applications. Also, the preference elicitation process of the MAUT approach requires users to correctly respond to the elicitation questions and to be consistent with the earlier tradeoffs they made. The same problem exists with tools that ask people to compare outcomes and then infer utility functions for these comparisons.

Boutilier et al. proposed a graphical representation of preferences, called CP-nets, that reflects conditional dependence and independence of preference statements under a ceteris paribus (all else being equal) interpretation[2]. CP-nets have the advantage of being able to efficiently represent the decision maker's preferences by the conditional preference statements which are rather natural to capture. However, being a qualitative method, the CP-net approach is unable to represent the quantitative information of users' preferences.

The framework of constraint satisfaction problems (CSPs) has been widely studied in AI research area for many years to solve different real-life problems ranging from map coloring, vision, robotics, VLSI design, etc[12][22]. Besides hard constraints that can never be violated, a CSP may also include soft constraints. These are functions that map any potential value assignment to a variable or combination of variables into a numerical value that indicates the preference that this value or value combination carries. CSPs with soft constraints provide a natural way of representing decision problems for which the user needs only to state the constraints of the problem to be modeled. Within the soft-CSP framework, each preference is modeled by one soft constraint, and the penalty function is determined as the sum of all the soft constraints. In this way it is easy for users to add, modify or remove part of their preferences without affecting the others. In this paper, we will represent users' various preferences based on the soft constraint technique and show how the agility requirements are satisfied.

3 Design of Agile Decision Agent

In this section, we first introduce the properties of user's preferences, and then present the example-critiquing interaction paradigm which has been proven effective in eliciting user's preferences. Finally we present the general system architecture of the agile decision agent in e-commerce environments.

3.1 Properties of User's Preferences

A user's preferences can be expressed in various forms. Based on our observation, we classify the user's preferences into the following 3 basic types:

1. *unit preference*. This kind of preferences only involves one attribute. An example of such a preference is:"*I'd like to depart at 10AM "*, where only the attribute $departure\_time$ is involved.

2. *conditional preference*. An example of such preferences is "*I'd like to depart at 10AM if the departure airport is Geneva*". This kind of preferences is effective only if the condition is valid.

3. *comparison preference*. This kind of preferences has the form of "I prefer item *A* to item *B*." In reality users are not likely to input such preferences into the system because product comparison demands much cognitive effort from the user.

In a real decision problem, the user may have both quantitative and qualitative preferences. The preference model should be able to represent all these types of preferences that the user may have. There may also have some *compound preferences* which involve more than one attribute or condition. But they can be easily decomposed into the combination of several basic ones. In this paper we will only discuss these basic preference types.

Tradeoff is an important aspect of preference construction as it is here that decision makers refine the details of their preferences into an accurate model. The representation of user's preferences should effectively support these tradeoffs and let user specify both qualitative and quantitative preferences at the same time. From observing user behaviors, we have identified the following 3 tradeoff strategies used by human decision makers:

- *value tradeoff*: a user explicitly changes the preference values of a set of attributes. In the soft constraint based framework, the user can change one of the soft constraints in the preference model, increasing or decreasing the penalty of a certain value combination in comparison with the others.

- *weight tradeoff*: a user changes the weights of a set of preferences so that those attributes with higher weight values take precedence when values are assigned. In a soft constraint based framework, the user can do this tradeoff directly.

- *outcome tradeoff*: a user performs either value or weight tradeoff after viewing a set of outcomes. In the soft constraint based framework, the user can provide additional soft constraints that modify the penalty of certain choices to make them acceptable.

Supporting these tradeoff strategies requires particular agility of the preference model that cannot be effectively supported by rigid elicitation procedures. The method of using soft constraints to model user's various preferences and to support preferences tradeoffs will be given in the next section.

3.2 Example Critiquing Interaction Paradigm

Preference construction must be supported by feedback from the system indicating the influence of the current model on the outcomes. A good way to implement such feedback is to structure user interaction as mixed-initiative systems (MISs). MISs are interactive problem solvers where human and machine intelligence are combined for their respective superiority[8]. MISs are therefore good candidates for such incremental decision systems.

A good way to implement a mixed-initiative decision support system is the *example critiquing* interaction paradigm as shown in Figure 1. It displays several examples of

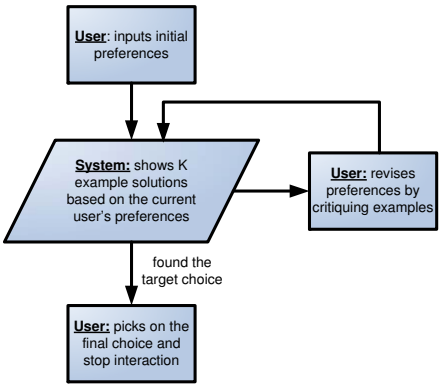

Figure 1. The example critiquing interaction paradigm.

complete solutions and invites users to state their critiques of these examples. Example critiquing allows users to better understand the impact of their preferences. Moreover, it provides an easy way for the user to add or revise his or her preferences at any time in any order during the decision making process. The user can input the type of *unit preferences* and *conditional preferences* directly through the user interfaces. The user may also input *comparison preferences* by selecting the preferred example from a list of candidates. Example-critiquing as an interface paradigm has been proposed by a variety of researchers[3][11][18]. The detail design method of example critiquing interaction has been studied in[6] and its performance has been evaluated in[15].

3.3 System Architecture

Figure 2 shows the general architecture of a service-oriented e-commerce system with the agile decision agent. The system can be divided into 3 layers: the web service infrastructure layer, the decision assistant layer and the user interface layer. The web service infrastructure layer is the underlying backbone of the service-oriented system. It contains the external atomic web services and a module of web service discovery and composition engine, which could obtain a list of items from the web services based on the hard constraints in the user's preference model and return them back to the decision agent.

The decision assistant layer is the key component that we address in this paper because it is here that we handle a user's preferences and dynamically generate a product list for the user to choose. Specifically, the decision agent consists of the following 3 modules:

- *Preference model*. The user's various preferences are maintained and converted to the proper form for the product search engine module. The user's preference model contains both hard and soft constraints. The detail of this model will be introduced in the next section.

- *Preference based product search engine*. In this module, we use the ranking mechanism provided by the weighted-CSP framework to generate the ranking list of all available items that the system may provide at the current scenario. This module is also responsible for extracting hard constraints from the user model and input them to the web services discover and composition engine to retrieve possible items dynamically during the interaction process.

- *Results generation*. It generates a list of top-K ranked items with minimal penalty values as feedback to the user. If some of the attributes violate the user's preferences, they will be highlighted with a special color (such as red) so the user can critique them in the next interaction cycle.

The user interface layer is designed to obtain the users' preferences and to show item information to users based on the example critiquing interaction paradigm, which enables users to construct their preferences incrementally according to the instant system feedback and thus avoid using fixed elicitation procedure which may lead to incorrect preference models. The agent is deployed in the server side of the system, and the interaction interface (be implemented by either HTML or java applet) is shown on internet browsers in the client side.

4 Agile Preference Modeling

4.1 The Soft-CSP Framework

A classical CSP[12][22] is characterized by a set of variables that can take values in associated domains and a set of constraints that define the allowed tuples for the variable or combination of variables. Solving a constraint satisfaction problem means finding one, several or all combinations of complete value assignments such that all the constraints are satisfied.

A classical CSP only contains hard constraints, that is, a constraint can be either satisfied or violated. Besides hard constraints, a CSP may also be extended to include soft constraints, which are functions that map any potential value assignment of a variable or a variable combination into a numerical value indicating the preference intension on this value or value combination. A CSP with soft constraints is called *soft-CSP* and solving a soft-CSP involves finding assignments that are optimally preferred with respect to the soft constraints.

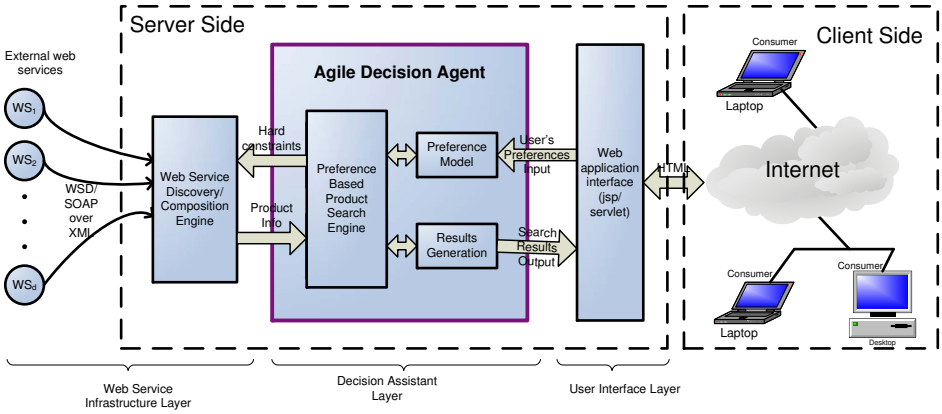

Figure 2. System architecture with the agile decision agent

There are various soft constraint formalisms that differ in the way the preference values of individual soft constraints are combined (combination function). For example, in weighted constraint satisfaction (weighted-CSP) [17], the optimal solution minimizes the weighted sum of the penalties of preferences. In fuzzy constraint satisfaction[16], the optimal solution maximizes the minimum preference value. Bistareli et al. [1] introduced a semiring-based CSP framework which can describe both classical and soft CSPs. Under this framework, a weighted-CSP can be represented by a c-semiring $\langle \Re, min, +, -\infty, \infty \rangle$.

4.2 Modeling Preferences as Soft Constraints

We apply the constraint satisfaction framework to multi-attribute decision problems by formulating each criterion as a separate constraint. The constraint programming framework replaces preference *elicitation*, a process of asking specific valuation questions, with preference *construction*, where users can manipulate individual criteria as separate constraints. In particular, it satisfies two important goals of agile preference models: (1) It allows users to precisely formulate their criteria without being restricted by the vocabulary of a database schema or elicitation tool; (2) Criteria can be added, removed or changed in any order they choose, as the model is not sensitive to an ordering among the preferences. In weighted-CSP, the possible items are ordered according to their penalty values and the item with the minimal penalty is regarded as the best choice. We adopt this mechanism to rank the items that a system may have as we found not only it can rank the items with only partial preferences, but also it corresponds best to users' intuition about the compensatory nature of tradeoffs.

Here we propose a preference based soft-CSP (PBSCSP) to model user's various preferences. Preferences that cannot be violated are represented as hard constraints to restrict the range of the possible items that the system may provide. For instance, if a user states that he or she would like to take a flight on a specific date for sure, then this information will be denoted as a hard constraint and transferred to the web service composition engine to retrieve all those possible flights satisfying this requirement. We use soft constraints to model the preferences stated by the user that can be violated for the purpose of tradeoff. We formally define the PBSCSP as the following.

Definition 1 (Preference-Based soft-CSP(PBSCSP)) *A PBSCSP is defined by a tuple* $\langle \boldsymbol{X}, \boldsymbol{D}, \Theta \rangle$*, where*

- $\boldsymbol{X} = \{x_1, x_2, ..., x_N\}$ *is a set of variables or attributes that the system may have;*
- $\boldsymbol{D} = \{D_1, D_2, ..., D_N\}$ *is a set of domains values for the corresponding attributes;*
- $\boldsymbol{\Theta}$ *is the user's preference model defined as below.*

Definition 2 (User's Preference Model (PM)) *A user's preference model* $\boldsymbol{\Theta}$ *is a set of* $\{\boldsymbol{C}, \widetilde{\boldsymbol{C}}, \boldsymbol{W}\}$*, where*

- $\boldsymbol{C} = \{C_1, C_2, ..., C_M\}$ *is a set of hard constraint functions that represent user's preferences which must be satisfied;*
- $\widetilde{\boldsymbol{C}} = \{\widetilde{C}_1, \widetilde{C}_2, ..., \widetilde{C}_{\widetilde{M}}\}$ *is a set of soft constraint functions that represent user's preferences that can potentially be violated for the tradeoff purpose;*
- $\boldsymbol{W} = \{w_1, w_2, ..., w_{\widetilde{M}}\}$ *is a set of weight values for the corresponding soft constraints.* $w_i (1 \leqslant i \leqslant \widetilde{M})$ *represents user's intensity of the preference represented by soft constraint* $\widetilde{C}_i$*.*

Based on a PM, we calculate the overall penalty value of a given item O as the following:

$$p(O) = \sum_{i=1}^{\widetilde{M}} w_i \widetilde{C}_i(O) \quad (1)$$

The user's preference model is required to be updated during the interaction process. When a user states a new preference, the preference conflict is first detected and any earlier conflict preferences are removed, and then the PM is extended by adding the newly stated preference.

4.3 The Interaction Procedure

Figure 3 shows the procedure for integrating example-critiquing interaction with the agile preference model applied in service-oriented e-commerce systems. First, the user's initial preferences (which mainly contain user's primary task goal) is obtained through the user interface and stored as the user preference model. Then the assistant agent fetches the available items which satisfy the hard constraints in the current user preference model through the module of web services discovery and composition engine. After that, the agent ranks the items based on the soft constraints in the current user preference model and selects as the candidate item set the top K items with minimal penalty values determined by equation 1. The candidate items are then shown to the user to elicit his or her more preferences. After getting the user's choice and the new preferences, the agent updates the user preference model and then repeats the interaction procedure until the user accepts the chosen item.

4.4 Example

As an example to illustrate how PBSCSP works, here we consider a travel planning system where there are 4 candidate items for choosing as shown in Table 1. A user may state the following 3 initial weighted criteria:

1. (weight = 1): I am willing to pay up to 1200$.:
   $\widetilde{C}_1 = price - 1200$

2. (weight = 100): The flight duration is desired to be 8 hours.:
   $\widetilde{C}_2 = duration - 8$

3. (weight = 50): I prefer intermediate airport Frankfurt over Zurich over Paris:
   $\widetilde{C}_3 = (\text{case Inter\_airport} : \begin{cases} Frankfurt : -2.0 \\ Zurich : -1.0 \\ Paris : 1.0 \end{cases})$

pm: user's preference model;
is: item set; *cs*: candidate set;
cp: current preferences; *p*: penalty value;

```
1.  procedure Example_Critiquing ( )
2.    pm = GetUserInitialPreferences ( )
3.    repeat
4.      is = FetchItems (pm)
5.      cs = GenerateCandidates (pm, is)
6.      choice = UserCritique (cs)
7.      cp = GetUserPreferences ( )
8.      pm = UpdateModel (pm, cp)
9.    until UserAccept (choice)

10. function GenerateCandidates (pm, is)
11.   cs = {}
12.   for each item O_i in is do
13.     p(O_i) = CalcPenalty(pm, O_i)
14.   end for
15.   is' = Sort_By_Penalty (is, p)
16.   cs = Top_K (is')
17.   return cs

18. function UpdateModel(pm, cp)
19.   pm = pm - Conflict(pm, cp)
20.   pm = pm + cp
21.   return pm
```

Figure 3. The interaction procedure between a user and the agile decision agent.

According to equation 1, the total penalty value of a candidate is a weighted sum of penalty values for these constraints. So the initial ranking assigned to the 4 outcomes would be:

$$\begin{aligned} p(O_1) &= 300 + 0 + 50 \times (-1.0) = 250 \\ p(O_2) &= 100 + 100 \times 1 + 50 \times (-2.0) = 100 \\ p(O_3) &= 200 + 100 \times 1 + 50 \times 1.0 = 350 \\ p(O_4) &= 0 + 100 \times 4 + 50 \times (-2.0) = 300 \end{aligned}$$

According to the penalty values, the outcomes can be ranked as: $O_2 \succ O_1 \succ O_4 \succ O_3$. However, in this case all outcomes are gained a penalty value, so the decision maker is not likely to pick any of them as a final solution.

Allowing users to state arbitrary criteria as soft constraints is a significant user interface challenge. We have found it sufficient to decide a certain number of useful parameterized criteria when designing the system and make these accessible through the user interface. While this limits the preferences that can be stated by a particular user,

	Price($)	Duration(h)	Inter_Airport	Dep_Time
O_1	1,500	8	Zurich	12:30PM
O_2	1,300	9	Frankfurt	9:30AM
O_3	1,400	9	Paris	10:45AM
O_4	1,200	12	Frankfurt	10:00AM

Table 1. Example: a list of several flight choices.

the framework itself places no limit on what criteria can be used for preferences.

Tradeoffs are required whenever none of the outcomes achieves a sufficiently high ranking according to the preference model. In the above example, the decision maker can engage in the three kinds of tradeoff mentioned earlier:

1. an example of a value tradeoff would be to change the desired value of $Duration$ from 8 to 9. Thus $\widetilde{C}_2$ is changed as "$\widetilde{C}_2 = duration - 9$" and O_2 would be selected in this case.

2. an example of a weight tradeoff would be to change the weight of intermediate airport preference from 50 to 100. Now choice O_2 has a penalty value of 0 and could be chosen as the target choice.

3. an example of an outcome tradeoff would be to add an additional attribute such as $Departure\_Time$ and then set different penalties for each value.

The required modification of the ranking is achieved by adding a preference or changing its weight in the combination model. In the soft constraint user preference model, these kinds of modifications are easy to carry out and provide an agile modeling process. Such agility would be very hard to achieve in models based on classical preference elicitation, where the impact of answering a particular query is hard to evaluate.

Note also that each of these three tradeoff types requires that the decision maker has a good understanding of the available outcomes and how they are affected by preferences. This understanding is provided, at least to some degree, by the example-critiquing framework: by showing examples that are close to the user's desires, it increases his or her understanding of the possibilities in the outcome space that are relevant to the desires.

5 Discussion

Multi-attribute utility theory (MAUT)[10] is a powerful tool for modeling preferences and rational decision-making. Let the symbol $\succsim$ denote the decision maker's preference order, e.g. $A \succsim B$ means "A is preferred or indifferent to B". According to MAUT, for a given MADP, there exists a function $U : \mathbf{O} \rightarrow \Re$, called a utility function, such that for any two possible items O_1 and O_2, we have $O_1 \succsim O_2 \Longleftrightarrow U(O_1) \geq U(O_2)$. Basing decisions on an explicit utility function forces the decision maker to reflect on the true preferences and can thus be expected to lead to more rational decisions and satisfying decisions.

The main difficulty with applying MAUT to preference-based search is how to elicit user preferences. Traditional methods for preference elicitation ask questions that assess the relative overall preference for certain attribute values. For example, when choosing a car, a typical question might be: "Which is better: 600 liters of trunk space and 4.5 meters minimum turning radius, or 1000 liters of trunk space and 5 meters of turning radius?" Answering such a question is not easy. It requires:

- imagining the different uses of the car, for example driving to work, buying groceries, and going on a family vacation. For the first two, manoeuvrability and thus the turning radius is important, whereas for the family vacation, trunk space is required.

- making the proper tradeoffs by considering how important the attributes are to the different uses, and which uses are more frequent.

- being consistent with earlier tradeoffs involving these attributes.

A user who is not very familiar with the domain is unlikely to be able to satisfy these requirements, and so his answers are not likely to reflect the correct utility function. Another problem is that a user may not have the patience to answer the many questions required by such an elicitation procedure.

On the other hand, in a model based on soft constraints, the preferences corresponding to the different uses of the car can be formulated as individual soft constraints. They can be expressed incrementally as the user becomes aware of her preferences, and they can easily be modified should she change her mind. Tradeoffs can be made explicitly at the level of conflicting preferences rather than separately for each attribute they involve, and it is thus easier to make them consistent. The user can choose the desired level of precision by formulating as many soft constraints as she likes.

This *agility* of the preference model supports the preference construction process and is thus likely to yield a more accurate model. The combination of soft constraints into a single ranking function can be understood as a *preference compiler* that constructs the multi-attribute utility function from the individual criteria while taking into account the relevant tradeoffs. In separate user studies, we have shown

the effectiveness of the approach in the context of example-critiquing decision aids in [15] and [14].

6 Conclusions and Future Work

Service-oriented computing provides structured data that makes it possible to automatically search for the most preferred item according to a user's preferences. Agility which allows consumers to add, retract and revise their preferences in any order and any time is an indispensable feature in service-oriented e-commerce systems. The contribution of this work is that we designed an agile decision agent to meet the agility requirement in the service-oriented e-commerce environment. We adopted the example-critiquing interaction paradigm to elicit user's diverse preferences, and proposed a soft-CSP (PBSCSP) model to represent those preferences and to rank those products dynamically. Though in this paper we discussed our design mainly in the travel planning domain, the agile decision agent itself is domain-independent. In the future, we plan to implement a travel planning system based on the proposed agile decision agent to test the agility performance through a set of real user studies. We will also optimize the overall performance of the system by more closely integrating the web service composition engine module and the decision agent to efficiently retrieve product information from multiple web services.

Acknowledgments

Funding for this research was provided by Swiss National Science Foundation under grant 200020-103490.

References

[1] S. Bistareli, H. Fargier, U. Montanari, F. Rossi, T. Schiex, and G. Verfaillie. Semiring-based CSPs and valued CSPs: Basic properties and comparison. *CONSTRAINTS: An International Journal*, 4(3), 1999.

[2] C. Boutilier, R. Brafman, C. Domshlak, H. Hoos, and D. Poole. CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *Journal of Artificial Intelligence Research*, 21:135–191, 2004.

[3] R. D. Burke, K. J. Hammond, and B. C. Young. The FindMe approach to assisted browsing. *IEEE Expert*, 12(4):32–40, 1997.

[4] G. Carenini and D. Poole. Constructed preferences and value-focused thinking: Implications for ai research on preference elicitation. In *Proceedings of the AAAI workshop on Preferences in AI and CP: Symbolic Approaches*, Edmonton, Canada, 2002. AAAI.

[5] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. W3c web services description language(WSDL) version 1.1, see http://www.w3.org/tr/2001/note-wsdl-20010315.

[6] B. Faltings, P. Pu, M. Torrens, and P. Viappiani. Designing example-critiquing interaction. In *Proceedings of the International Conference on Intelligent User Interfaces*, pages 22–29, Island of Madeira (Portugal), January 2004. ACM.

[7] G. Haubl and V. Trifts. Consumer decision making in online shopping environments: The effects of interactive decision aids. *Marketing Science*, 19(1):4–21, 2000.

[8] E. Horvitz. Principles of mixed-initiative user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems(CHI'99)*, pages 159–166. ACM Press, 1999.

[9] R. Keeney. *Value-Focused Thinking: A Path to Creative Decision Making*. Harvard University Press, Cambridge, 1992.

[10] R. Keeney and H. Raiffa. *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*. John Wiley and Sons, New York, 1976.

[11] G. Linden, S. Hanks, and N. Lesh. Interactive assessment of user preference models: The automated travel assistant. In *Proceedings of the 6th International Conference on User Modeling (UM97)*, 1997.

[12] A. Mackworth. Constraint satisfaction. *Encyclopedia of Artificial Intelligence*, pages 205–211, 1988.

[13] J. Payne, J. Bettman, and E. Johnson. *The Adaptive Decision Maker*. Cambridge University Press, 1993.

[14] P. Pu and L. Chen. Integrating tradeoff support in product search tools for e-commerce sits. In *Proceedings of the ACM Conference on Electronic Commerce (EC'05)*, pages 269–278, Vancouver, Canada, 2005.

[15] P. Pu and P. Kumar. Evaluating example-based search tools. In *Proceedings of the ACM Conference on Electronic Commerce (EC'04)*, pages 208–217, New York, USA, 2004.

[16] Z. Ruttkay. Fuzzy constraint satisfaction. In *Proceedings of the 3rd IEEE International Conference on Fuzzy Systems*, pages 1263–1268, 1994.

[17] T. Schiex, H. Fargier, and G. Verfaillie. Valued constraint satisfaction problems: Hard and easy problems. In *Proceedings International Joint Conference on Artificial Intelligence(IJCAI'95):*, pages 631–639, 1995.

[18] S. Shearin and H. Lieberman. Intelligent profiling by example. In *Proceedings of the Conference on Intelligent User Interfaces*, pages 145–151. ACM Press New York, NY, USA, 2001.

[19] H. Shimazu. Expertclerk: Navigating shoppers buying process with the combination of asking and proposing. In *Proceedings of the 17 International Joint Conference on Artificial Intelligence (IJCAI'01)*, volume 2, pages 1443–1448, Seattle, Washington, USA, 2001.

[20] H. A. Simon. A behavioral model of rational choice. *Quarterly Journal of Economics*, 69:99–118, 1955.

[21] M. Stolze. Soft navigation in electronic product catalogs. *International Journal on Digital Libraries*, 3(1):60–66, July 2000.

[22] E. Tsang. *Foundations of Constraint Satisfaction*. Academic Press, London, UK, 1993.